Assignment Report: CSC3150 Assignment 3

Wangmeiyu Zhang - 123090825

All the screen shoots (testcase output) can be found under the source code folder 'screen-shoots'.

The test file is 'Q3_test.c' (same as 'Q3_test.c' under 'user' folder). You can try other test cases using cpu_worker(), io_worker() and mixed_worker().

The comments highlighting key changes has been added into the source code with 'DONE.'.

1 Introduction [2']

This assignment implements a 3-level priority queue scheduler for the xv6-riscv operating system. The original xv6 uses a simple round-robin scheduler, which this assignment enhances by introducing multiple priority queues with different time slices and anti-starvation mechanisms. New processes start in the highest priority queue Q0; processes that use their entire time slice are demoted to the next lower queue; processes that block on I/O return to their original queue when woken; every 100 ticks, all processes in Q1 and Q2 are promoted to Q0 to prevent starvation; and processes within the same queue are scheduled in a Round-Robin manner. The implementation involved significant modifications to the process scheduling subsystem, including data structure changes, timer interrupt handling, and the introduction of new scheduling policies.

The modifications were made to `proc.c`, `proc.h`, `trap.c`, `syscall.c`, `syscall.h`, `defs.h`, `sysproc.c`, `user.h`, `usys.pl`, `Makefile`, and the test program `Q3_test.c` was modified to validate the implementation. The new scheduler effectively balances responsiveness for interactive processes while maintaining fairness through periodic promotions.

2 Design [5']

2.1 Scheduling Algorithm Design

The 3-level priority queue scheduler operates on the following principles:

- Three Priority Queues: (Modified a little bit for better performance) Newly created processes start in Q0. Each queue is implemented using link list.
  - Q0: Highest priority with shortest time slice (6 ticks)
  - Q1: Medium priority with medium time slice (12 ticks)
  - Q2: Lowest priority with longest time slice (24 ticks)
- Process Migration: On each clock interrupt, the currently running process's time slice is decremented. Processes that use their entire time slice are demoted to the next lower queue (Q0 $\rightarrow$ Q1 $\rightarrow$ Q2), what in Q2 will still in Q2 after demotion.
- I/O Handling: If a process voluntarily relinquishes the CPU (e.g., by calling `sleep()`), it returns to its original queue upon waking.
- Anti-Starvation: Every 100 ticks, all processes (except sleeping processed) in Q1 and Q2 are promoted to Q0. This project check all processed in queue by travling the link list, which means promotion will not affect the relative order. For RUNNING processes, this project change its level to 0 but not add it into the queue (since it

is not in the RUNNABLE queue when RUNNING). In this way, it could go to Q0 after RUNNING.

- Intra-queue Scheduling: Processes within the same queue are scheduled using round-robin

2.2 Key Data Structures

In `proc.h`, the process control block (`struct proc`) is extended with the following fields:

```
struct proc *next_in_queue; // next proc in queue
int queue_level;         // (0,1,2)
int remaining_ticks;     // for one proc. in this queue level
int original_queue;      // be used for wake()
```

In `proc.c`, three queues (linked lists) are defined to hold ready processes. The `scheduler()` function checks these queues in order from highest to lowest priority for runnable processes.

2.3 Algorithm Implementation

The scheduling algorithm was implemented through several key functions:

1. Process Initialization: New processes start in Q0 with 3-tick time slice
2. Scheduler Loop: The scheduler selects processes from highest to lowest priority queue
3. Time Slice Management: Timer interrupts track time slice usage and trigger queue transitions
4. Promotion Mechanism: Global timer tracks 100-tick intervals for anti-starvation promotions

2.4 Key Code Modifications

2.4.1 Scheduler Implementation (`proc.c`)

Modified the scheduler to implement multi-level queue scheduling:

```
void scheduler(void) {
  for(;;) {
    intr_on();
    intr_off();// prevent dead lock


    acquire(&promote_lock);//using lock
    if (promote_needed) {
      promote_needed = 0;// global variable
      release(&promote_lock);
      promote_processes();//promote all processed in Q1 and Q2 to Q0
    } else {
      release(&promote_lock);
    }


    // Check each priority queue from highest to lowest
    for(int level = 0; level < NQUEUE && !found; level++) {
        p = dequeue_proc(level);
        //Does not acquire and release queue lock,
        //since it works for dequeue and has implemented in dequeue_proc()
        if(p != 0) {
            // Switch to chosen process
            acquire(&p->lock);
            if(p->state == RUNNABLE) {
                p->state = RUNNING;
                c->proc = p;
                swtch(&c->context, &p->context);

                c->proc = 0;
                if(p->state == RUNNABLE) {
                    enqueue_proc(p, p->queue_level);
                }
                found = 1;
                release(&p->lock);
            }
        }
    }

    if(found == 0) {
      // nothing to run; stop running on this core until an interrupt.
      asm volatile("wfi");
    }
  }
}
```

2.4.2 Timer Interrupt Handling (**trap.c**)

Enhanced timer interrupt handling to manage time slices and queue promotions:

```
void usertrap(void) {
  // ...
    if(which_dev == 2){
        // Timer interrupt
        struct proc *p = myproc();
        if(p != 0 && p->state == RUNNING) {
            acquire(&p->lock);
            if(p->remaining_ticks > 0) {
                p->remaining_ticks--;
            }
            // Check if process used its entire time slice
            if(p->remaining_ticks <= 0) {
                // Demote process to next lower queue
                if(p->queue_level < 2) {
                    p->queue_level++;
                }
                switch(p->queue_level) {
                    case 0: p->remaining_ticks = Q0_TICKS; break;
                    case 1: p->remaining_ticks = Q1_TICKS; break;
                    case 2: p->remaining_ticks = Q2_TICKS; break;
                }
            }
            release(&p->lock);
        }
        yield();
    }
  // ...
}
```

2.4.3 System Call for Testing (**sysproc.c**)

Added a system call to allow user programs to query their current queue:

```
uint64
```

```
sys_getpriority(void)
{
  struct proc *p = myproc();
  return p->queue_level;
}
```

2.4.4 Others

1. `proc.c`: In `allocproc`, new processes are initialized with priority 0 (Q0) and a time slice. Locks are initialized. Queue operations are done in `promote_processes`, `remove_from_ queue`,`dequeue_proc` and `enqueue_proc`.For system calls like `sleep`, the original queue is recorded when the process blocks and restored upon wake-up.
2. `trap.c`: In `clockintr`, every 100 ticks, global variable is set for calling to move all Q1 and Q2 processes to Q0.
3. `Q3_test.c`: A test program is written to create multiple CPU-intensive, I/O-intensive and Mix processes, observing their migration between queues.

3 Environment and Execution [2']

3.1 Running Environment

The experiment is conducted on xv6-riscv using the QEMU emulator. The compilation environment is Ubuntu 20.04. Multi kernel CPU.

3.2 Execution Instructions

```
$ make clean
$ make qemu
```

To run the test program inside xv6:

```
$ Q3_test
```

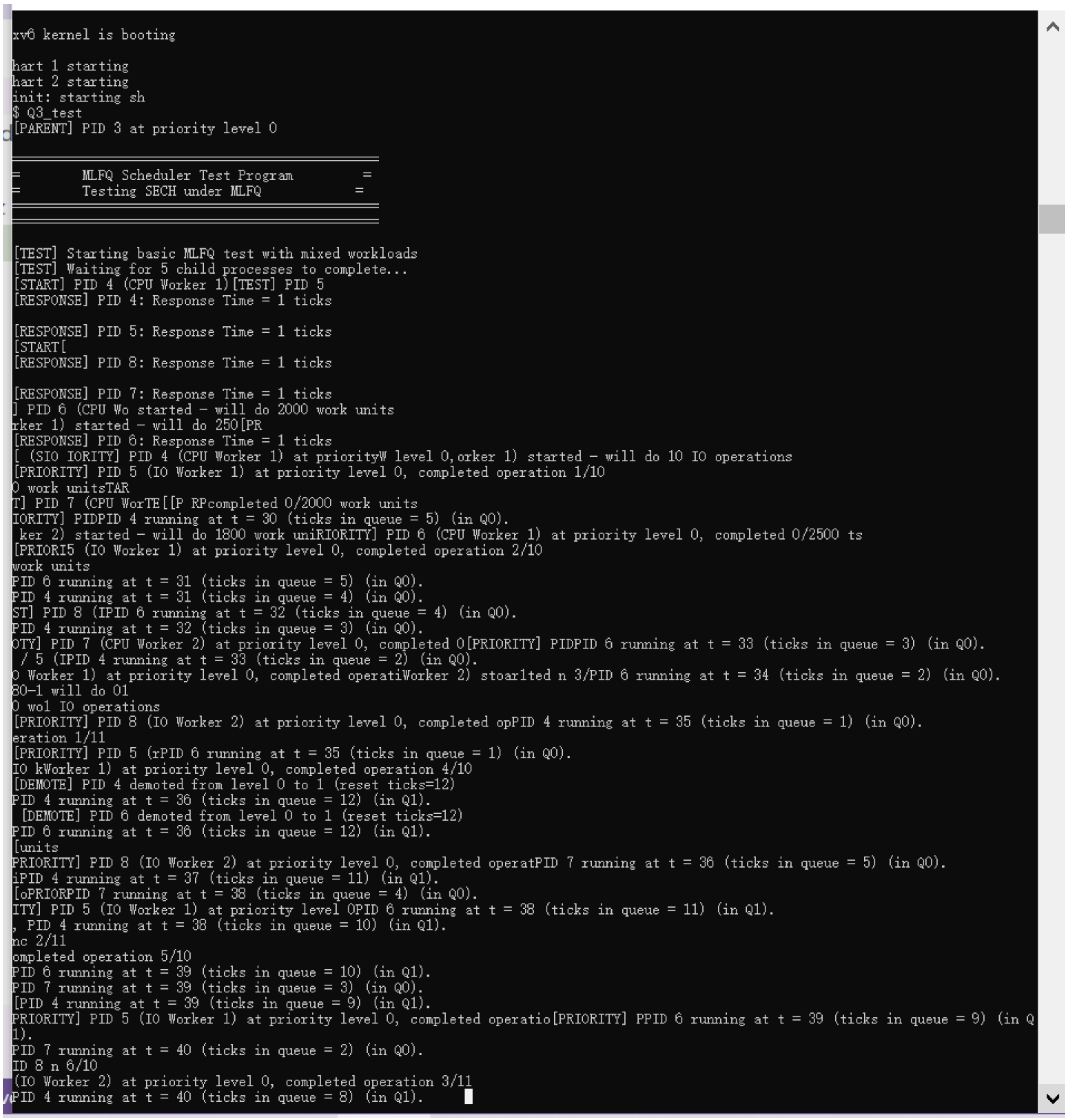
If you want to print informations (as shown in pictures), please set 'MLFQ_DEBUG' in `proc.c`, `trap.c`, and `Q3_test.c` as 1 (as the submitted codes). Otherwise, please those to 0 (i.e. '#define MLFQ_DEBUG 0' rather than '#define MLFQ_DEBUG 1'). Then this project will show defalut information as given Q3_test.c template, 'Throughput' and 'Response'.

3.3 Test Cases and Expected Output

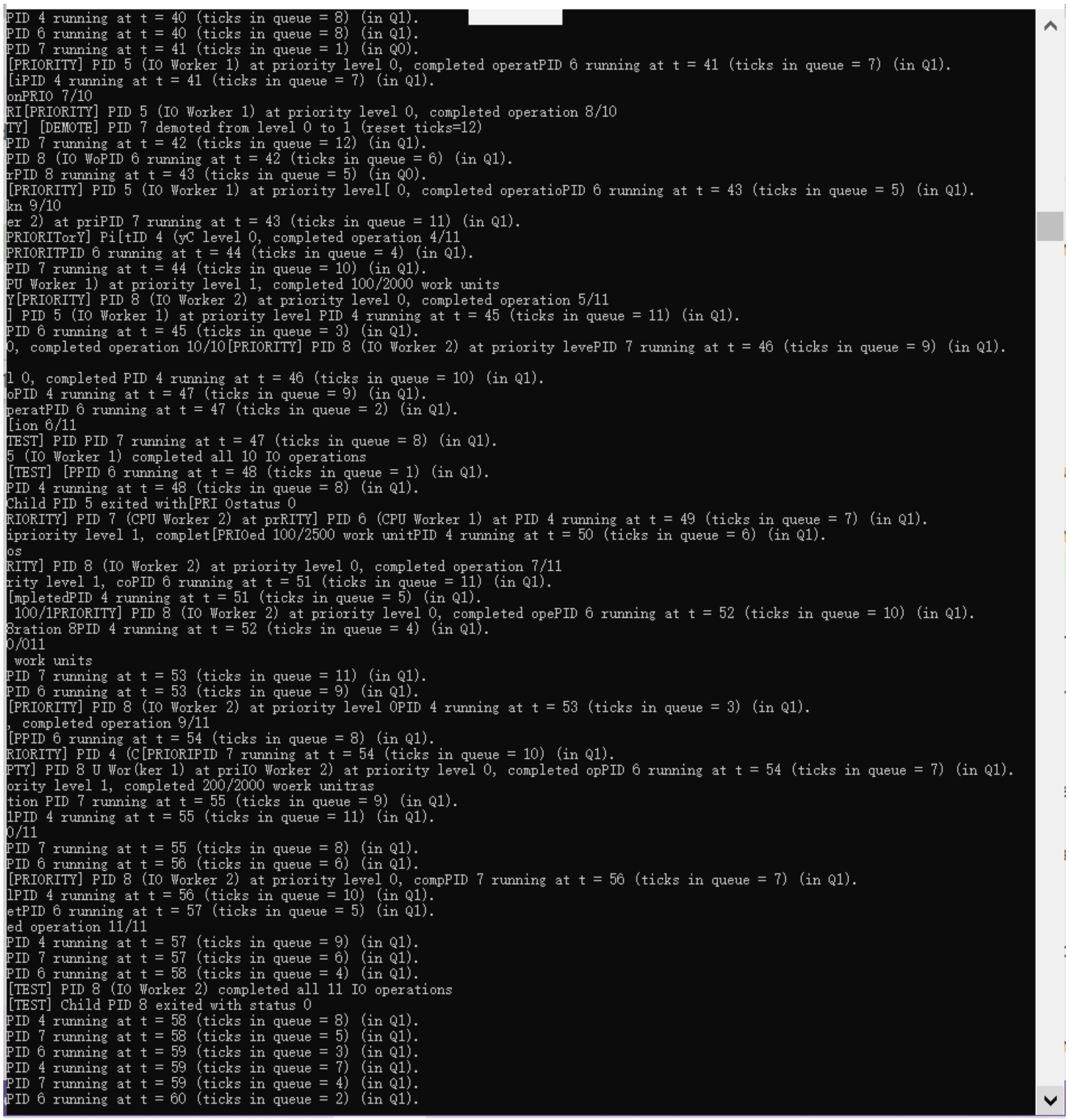
The test program creates three types of processes:

- CPU-intensive: Long CPU usage, gradually demoted from Q0 to Q2. Promoted to Q0 every 100 ticks.
- I/O-intensive: Frequent `sleep()` calls, mostly remaining in Q0.
- Mixed: Alternating CPU and I/O usage.

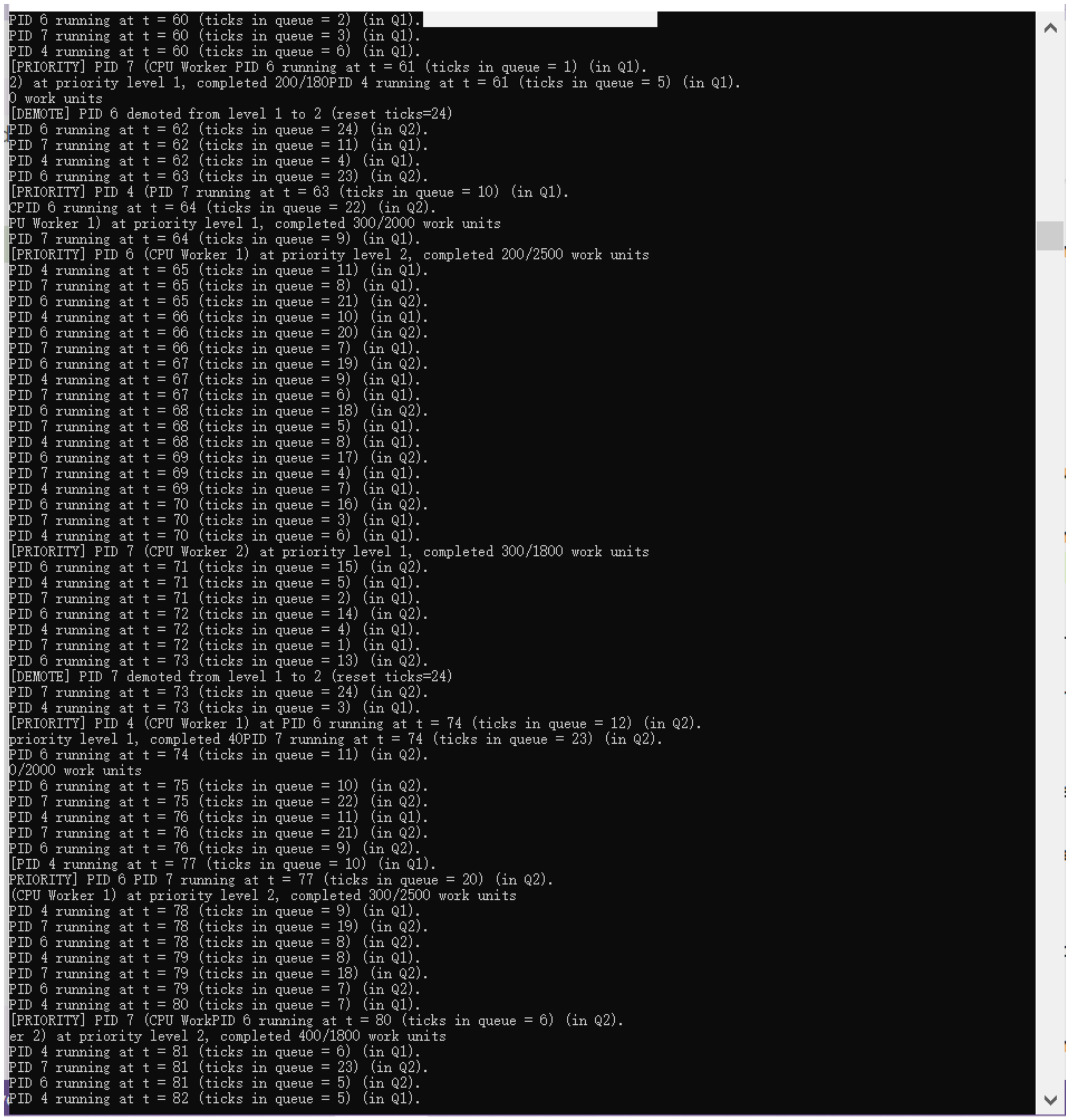
For simplify, this report only shows key outputs. You can find the whole screen shoots under the source code floder.



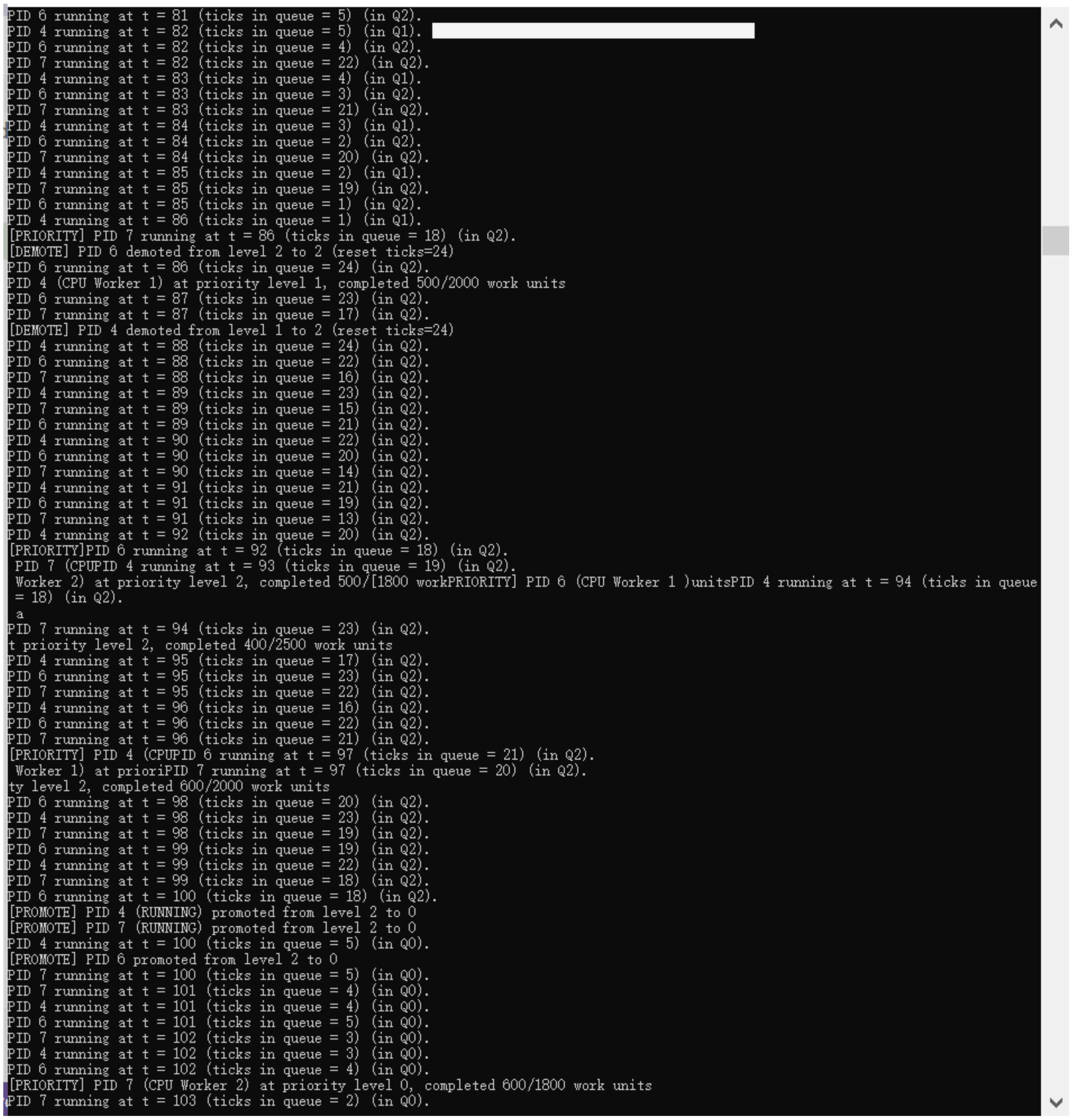
```
xv6 kernel is booting

hart 1 starting
hart 2 starting
init: starting sh
$ Q3_test
[PARENT] PID 3 at priority level 0

==============================================
=        MLFQ Scheduler Test Program         =
=        Testing SECH under MLFQ            =
==============================================
==============================================


[TEST] Starting basic MLFQ test with mixed workloads
[TEST] Waiting for 5 child processes to complete...
[START] PID 4 (CPU Worker 1)[TEST] PID 5
[RESPONSE] PID 4: Response Time = 1 ticks

[RESPONSE] PID 5: Response Time = 1 ticks
[START[
[RESPONSE] PID 8: Response Time = 1 ticks

[RESPONSE] PID 7: Response Time = 1 ticks
] PID 6 (CPU Wo started - will do 2000 work units
rker 1) started - will do 250[PR
[RESPONSE] PID 6: Response Time = 1 ticks
[ (SIO IORITY] PID 4 (CPU Worker 1) at priorityW level 0,orker 1) started - will do 10 IO operations
[PRIORITY] PID 5 (IO Worker 1) at priority level 0, completed operation 1/10
0 work unitsTAR
T] PID 7 (CPU WorTE[[P RPcompleted 0/2000 work units
IORITY] PIDPID 4 running at t = 30 (ticks in queue = 5) (in Q0).
 ker 2) started - will do 1800 work uniRIORITY] PID 6 (CPU Worker 1) at priority level 0, completed 0/2500 ts
[PRIORI5 (IO Worker 1) at priority level 0, completed operation 2/10
work units
PID 6 running at t = 31 (ticks in queue = 5) (in Q0).
PID 4 running at t = 31 (ticks in queue = 4) (in Q0).
ST] PID 8 (IPID 6 running at t = 32 (ticks in queue = 4) (in Q0).
PID 4 running at t = 32 (ticks in queue = 3) (in Q0).
OTY] PID 7 (CPU Worker 2) at priority level 0, completed 0[PRIORITY] PIDPID 6 running at t = 33 (ticks in queue = 3) (in Q0).
 / 5 (IPID 4 running at t = 33 (ticks in queue = 2) (in Q0).
0 Worker 1) at priority level 0, completed operatiWorker 2) stoar1ted n 3/PID 6 running at t = 34 (ticks in queue = 2) (in Q0).
80-1 will do 01
0 wo1 IO operations
[PRIORITY] PID 8 (IO Worker 2) at priority level 0, completed opPID 4 running at t = 35 (ticks in queue = 1) (in Q0).
eration 1/11
[PRIORITY] PID 5 (rPID 6 running at t = 35 (ticks in queue = 1) (in Q0).
IO kWorker 1) at priority level 0, completed operation 4/10
[DEMOTE] PID 4 demoted from level 0 to 1 (reset ticks=12)
PID 4 running at t = 36 (ticks in queue = 12) (in Q1).
 [DEMOTE] PID 6 demoted from level 0 to 1 (reset ticks=12)
PID 6 running at t = 36 (ticks in queue = 12) (in Q1).
[units
PRIORITY] PID 8 (IO Worker 2) at priority level 0, completed operatPID 7 running at t = 36 (ticks in queue = 5) (in Q0).
iPID 4 running at t = 37 (ticks in queue = 11) (in Q1).
[oPRIORPID 7 running at t = 38 (ticks in queue = 4) (in Q0).
ITY] PID 5 (IO Worker 1) at priority level 0PID 6 running at t = 38 (ticks in queue = 11) (in Q1).
, PID 4 running at t = 38 (ticks in queue = 10) (in Q1).
nc 2/11
ompleted operation 5/10
PID 6 running at t = 39 (ticks in queue = 10) (in Q1).
PID 7 running at t = 39 (ticks in queue = 3) (in Q0).
[PID 4 running at t = 39 (ticks in queue = 9) (in Q1).
PRIORITY] PID 5 (IO Worker 1) at priority level 0, completed operatio[PRIORITY] PPID 6 running at t = 39 (ticks in queue = 9) (in Q
1).
PID 7 running at t = 40 (ticks in queue = 2) (in Q0).
ID 8 n 6/10
(IO Worker 2) at priority level 0, completed operation 3/11
PID 4 running at t = 40 (ticks in queue = 8) (in Q1).
```



```
PID 4 running at t = 40 (ticks in queue = 8) (in Q1).
PID 6 running at t = 40 (ticks in queue = 8) (in Q1).
PID 7 running at t = 41 (ticks in queue = 1) (in Q0).
[PRIORITY] PID 5 (IO Worker 1) at priority level 0, completed operatPID 6 running at t = 41 (ticks in queue = 7) (in Q1).
[iPID 4 running at t = 41 (ticks in queue = 7) (in Q1).
onPRIO 7/10
RI[PRIORITY] PID 5 (IO Worker 1) at priority level 0, completed operation 8/10
TY] [DEMOTE] PID 7 demoted from level 0 to 1 (reset ticks=12)
PID 7 running at t = 42 (ticks in queue = 12) (in Q1).
PID 8 (IO WoPID 6 running at t = 42 (ticks in queue = 6) (in Q1).
rPID 8 running at t = 43 (ticks in queue = 5) (in Q0).
[PRIORITY] PID 5 (IO Worker 1) at priority level[ 0, completed operatioPID 6 running at t = 43 (ticks in queue = 5) (in Q1).
kn 9/10
er 2) at priPID 7 running at t = 43 (ticks in queue = 11) (in Q1).
PRIORITorY] Pi[tID 4 (yC level 0, completed operation 4/11
PRIORITPID 6 running at t = 44 (ticks in queue = 4) (in Q1).
PID 7 running at t = 44 (ticks in queue = 10) (in Q1).
PU Worker 1) at priority level 1, completed 100/2000 work units
Y[PRIORITY] PID 8 (IO Worker 2) at priority level 0, completed operation 5/11
] PID 5 (IO Worker 1) at priority level PID 4 running at t = 45 (ticks in queue = 11) (in Q1).
PID 6 running at t = 45 (ticks in queue = 3) (in Q1).
0, completed operation 10/10[PRIORITY] PID 8 (IO Worker 2) at priority levePID 7 running at t = 46 (ticks in queue = 9) (in Q1).

l 0, completed PID 4 running at t = 46 (ticks in queue = 10) (in Q1).
oPID 4 running at t = 47 (ticks in queue = 9) (in Q1).
peratPID 6 running at t = 47 (ticks in queue = 2) (in Q1).
[ion 6/11
TEST] PID PID 7 running at t = 47 (ticks in queue = 8) (in Q1).
5 (IO Worker 1) completed all 10 IO operations
[TEST] [PPID 6 running at t = 48 (ticks in queue = 1) (in Q1).
PID 4 running at t = 48 (ticks in queue = 8) (in Q1).
Child PID 5 exited with[PRI 0status 0
RIORITY] PID 7 (CPU Worker 2) at prRITY] PID 6 (CPU Worker 1) at PID 4 running at t = 49 (ticks in queue = 7) (in Q1).
ipriority level 1, complet[PRIOed 100/2500 work unitPID 4 running at t = 50 (ticks in queue = 6) (in Q1).
os
RITY] PID 8 (IO Worker 2) at priority level 0, completed operation 7/11
rity level 1, coPID 6 running at t = 51 (ticks in queue = 11) (in Q1).
[mpletedPID 4 running at t = 51 (ticks in queue = 5) (in Q1).
 100/1PRIORITY] PID 8 (IO Worker 2) at priority level 0, completed opePID 6 running at t = 52 (ticks in queue = 10) (in Q1).
8ration 8PID 4 running at t = 52 (ticks in queue = 4) (in Q1).
0/011
 work units
PID 7 running at t = 53 (ticks in queue = 11) (in Q1).
PID 6 running at t = 53 (ticks in queue = 9) (in Q1).
[PRIORITY] PID 8 (IO Worker 2) at priority level 0PID 4 running at t = 53 (ticks in queue = 3) (in Q1).
, completed operation 9/11
[PPID 6 running at t = 54 (ticks in queue = 8) (in Q1).
RIORITY] PID 4 (C[PRIORIPID 7 running at t = 54 (ticks in queue = 10) (in Q1).
PTY] PID 8 U Wor(ker 1) at priIO Worker 2) at priority level 0, completed opPID 6 running at t = 54 (ticks in queue = 7) (in Q1).
ority level 1, completed 200/2000 woerk unitras
tion PID 7 running at t = 55 (ticks in queue = 9) (in Q1).
1PID 4 running at t = 55 (ticks in queue = 11) (in Q1).
0/11
PID 7 running at t = 55 (ticks in queue = 8) (in Q1).
PID 6 running at t = 56 (ticks in queue = 6) (in Q1).
[PRIORITY] PID 8 (IO Worker 2) at priority level 0, compPID 7 running at t = 56 (ticks in queue = 7) (in Q1).
lPID 4 running at t = 56 (ticks in queue = 10) (in Q1).
etPID 6 running at t = 57 (ticks in queue = 5) (in Q1).
ed operation 11/11
PID 4 running at t = 57 (ticks in queue = 9) (in Q1).
PID 7 running at t = 57 (ticks in queue = 6) (in Q1).
PID 6 running at t = 58 (ticks in queue = 4) (in Q1).
[TEST] PID 8 (IO Worker 2) completed all 11 IO operations
[TEST] Child PID 8 exited with status 0
PID 4 running at t = 58 (ticks in queue = 8) (in Q1).
PID 7 running at t = 58 (ticks in queue = 5) (in Q1).
PID 6 running at t = 59 (ticks in queue = 3) (in Q1).
PID 4 running at t = 59 (ticks in queue = 7) (in Q1).
PID 7 running at t = 59 (ticks in queue = 4) (in Q1).
PID 6 running at t = 60 (ticks in queue = 2) (in Q1).
```



```
PID 6 running at t = 60 (ticks in queue = 2) (in Q1).
PID 7 running at t = 60 (ticks in queue = 3) (in Q1).
PID 4 running at t = 60 (ticks in queue = 6) (in Q1).
[PRIORITY] PID 7 (CPU Worker PID 6 running at t = 61 (ticks in queue = 1) (in Q1).
2) at priority level 1, completed 200/180PID 4 running at t = 61 (ticks in queue = 5) (in Q1).
0 work units
[DEMOTE] PID 6 demoted from level 1 to 2 (reset ticks=24)
PID 6 running at t = 62 (ticks in queue = 24) (in Q2).
PID 7 running at t = 62 (ticks in queue = 11) (in Q1).
PID 4 running at t = 62 (ticks in queue = 4) (in Q1).
PID 6 running at t = 63 (ticks in queue = 23) (in Q2).
[PRIORITY] PID 4 (PID 7 running at t = 63 (ticks in queue = 10) (in Q1).
CPID 6 running at t = 64 (ticks in queue = 22) (in Q2).
PU Worker 1) at priority level 1, completed 300/2000 work units
PID 7 running at t = 64 (ticks in queue = 9) (in Q1).
[PRIORITY] PID 6 (CPU Worker 1) at priority level 2, completed 200/2500 work units
PID 4 running at t = 65 (ticks in queue = 11) (in Q1).
PID 7 running at t = 65 (ticks in queue = 8) (in Q1).
PID 6 running at t = 65 (ticks in queue = 21) (in Q2).
PID 4 running at t = 66 (ticks in queue = 10) (in Q1).
PID 6 running at t = 66 (ticks in queue = 20) (in Q2).
PID 7 running at t = 66 (ticks in queue = 7) (in Q1).
PID 6 running at t = 67 (ticks in queue = 19) (in Q2).
PID 4 running at t = 67 (ticks in queue = 9) (in Q1).
PID 7 running at t = 67 (ticks in queue = 6) (in Q1).
PID 6 running at t = 68 (ticks in queue = 18) (in Q2).
PID 7 running at t = 68 (ticks in queue = 5) (in Q1).
PID 4 running at t = 68 (ticks in queue = 8) (in Q1).
PID 6 running at t = 69 (ticks in queue = 17) (in Q2).
PID 7 running at t = 69 (ticks in queue = 4) (in Q1).
PID 4 running at t = 69 (ticks in queue = 7) (in Q1).
PID 6 running at t = 70 (ticks in queue = 16) (in Q2).
PID 7 running at t = 70 (ticks in queue = 3) (in Q1).
PID 4 running at t = 70 (ticks in queue = 6) (in Q1).
[PRIORITY] PID 7 (CPU Worker 2) at priority level 1, completed 300/1800 work units
PID 6 running at t = 71 (ticks in queue = 15) (in Q2).
PID 4 running at t = 71 (ticks in queue = 5) (in Q1).
PID 7 running at t = 71 (ticks in queue = 2) (in Q1).
PID 6 running at t = 72 (ticks in queue = 14) (in Q2).
PID 4 running at t = 72 (ticks in queue = 4) (in Q1).
PID 7 running at t = 72 (ticks in queue = 1) (in Q1).
PID 6 running at t = 73 (ticks in queue = 13) (in Q2).
[DEMOTE] PID 7 demoted from level 1 to 2 (reset ticks=24)
PID 7 running at t = 73 (ticks in queue = 24) (in Q2).
PID 4 running at t = 73 (ticks in queue = 3) (in Q1).
[PRIORITY] PID 4 (CPU Worker 1) at PID 6 running at t = 74 (ticks in queue = 12) (in Q2).
priority level 1, completed 40PID 7 running at t = 74 (ticks in queue = 23) (in Q2).
PID 6 running at t = 74 (ticks in queue = 11) (in Q2).
0/2000 work units
PID 6 running at t = 75 (ticks in queue = 10) (in Q2).
PID 7 running at t = 75 (ticks in queue = 22) (in Q2).
PID 4 running at t = 76 (ticks in queue = 11) (in Q1).
PID 7 running at t = 76 (ticks in queue = 21) (in Q2).
PID 6 running at t = 76 (ticks in queue = 9) (in Q2).
[PID 4 running at t = 77 (ticks in queue = 10) (in Q1).
PRIORITY] PID 6 PID 7 running at t = 77 (ticks in queue = 20) (in Q2).
(CPU Worker 1) at priority level 2, completed 300/2500 work units
PID 4 running at t = 78 (ticks in queue = 9) (in Q1).
PID 7 running at t = 78 (ticks in queue = 19) (in Q2).
PID 6 running at t = 78 (ticks in queue = 8) (in Q2).
PID 4 running at t = 79 (ticks in queue = 8) (in Q1).
PID 7 running at t = 79 (ticks in queue = 18) (in Q2).
PID 6 running at t = 79 (ticks in queue = 7) (in Q2).
PID 4 running at t = 80 (ticks in queue = 7) (in Q1).
[PRIORITY] PID 7 (CPU WorkPID 6 running at t = 80 (ticks in queue = 6) (in Q2).
er 2) at priority level 2, completed 400/1800 work units
PID 4 running at t = 81 (ticks in queue = 6) (in Q1).
PID 7 running at t = 81 (ticks in queue = 23) (in Q2).
PID 6 running at t = 81 (ticks in queue = 5) (in Q2).
PID 4 running at t = 82 (ticks in queue = 5) (in Q1).
```



```
PID 6 running at t = 81 (ticks in queue = 5) (in Q2).
PID 4 running at t = 82 (ticks in queue = 5) (in Q1).
PID 6 running at t = 82 (ticks in queue = 4) (in Q2).
PID 7 running at t = 82 (ticks in queue = 22) (in Q2).
PID 4 running at t = 83 (ticks in queue = 4) (in Q1).
PID 6 running at t = 83 (ticks in queue = 3) (in Q2).
PID 7 running at t = 83 (ticks in queue = 21) (in Q2).
PID 4 running at t = 84 (ticks in queue = 3) (in Q1).
PID 6 running at t = 84 (ticks in queue = 2) (in Q2).
PID 7 running at t = 84 (ticks in queue = 20) (in Q2).
PID 4 running at t = 85 (ticks in queue = 2) (in Q1).
PID 7 running at t = 85 (ticks in queue = 19) (in Q2).
PID 6 running at t = 85 (ticks in queue = 1) (in Q2).
PID 4 running at t = 86 (ticks in queue = 1) (in Q1).
[PRIORITY] PID 7 running at t = 86 (ticks in queue = 18) (in Q2).
[DEMOTE] PID 6 demoted from level 2 to 2 (reset ticks=24)
PID 6 running at t = 86 (ticks in queue = 24) (in Q2).
PID 4 (CPU Worker 1) at priority level 1, completed 500/2000 work units
PID 6 running at t = 87 (ticks in queue = 23) (in Q2).
PID 7 running at t = 87 (ticks in queue = 17) (in Q2).
[DEMOTE] PID 4 demoted from level 1 to 2 (reset ticks=24)
PID 4 running at t = 88 (ticks in queue = 24) (in Q2).
PID 6 running at t = 88 (ticks in queue = 22) (in Q2).
PID 7 running at t = 88 (ticks in queue = 16) (in Q2).
PID 4 running at t = 89 (ticks in queue = 23) (in Q2).
PID 7 running at t = 89 (ticks in queue = 15) (in Q2).
PID 6 running at t = 89 (ticks in queue = 21) (in Q2).
PID 4 running at t = 90 (ticks in queue = 22) (in Q2).
PID 6 running at t = 90 (ticks in queue = 20) (in Q2).
PID 7 running at t = 90 (ticks in queue = 14) (in Q2).
PID 4 running at t = 91 (ticks in queue = 21) (in Q2).
PID 6 running at t = 91 (ticks in queue = 19) (in Q2).
PID 7 running at t = 91 (ticks in queue = 13) (in Q2).
PID 4 running at t = 92 (ticks in queue = 20) (in Q2).
[PRIORITY]PID 6 running at t = 92 (ticks in queue = 18) (in Q2).
 PID 7 (CPUPID 4 running at t = 93 (ticks in queue = 19) (in Q2).
 Worker 2) at priority level 2, completed 500/[1800 workPRIORITY] PID 6 (CPU Worker 1 )unitsPID 4 running at t = 94 (ticks in queue
 = 18) (in Q2).
 a
PID 7 running at t = 94 (ticks in queue = 23) (in Q2).
t priority level 2, completed 400/2500 work units
PID 4 running at t = 95 (ticks in queue = 17) (in Q2).
PID 6 running at t = 95 (ticks in queue = 23) (in Q2).
PID 7 running at t = 95 (ticks in queue = 22) (in Q2).
PID 4 running at t = 96 (ticks in queue = 16) (in Q2).
PID 6 running at t = 96 (ticks in queue = 22) (in Q2).
PID 7 running at t = 96 (ticks in queue = 21) (in Q2).
[PRIORITY] PID 4 (CPUPID 6 running at t = 97 (ticks in queue = 21) (in Q2).
 Worker 1) at prioriPID 7 running at t = 97 (ticks in queue = 20) (in Q2).
ty level 2, completed 600/2000 work units
PID 6 running at t = 98 (ticks in queue = 20) (in Q2).
PID 4 running at t = 98 (ticks in queue = 23) (in Q2).
PID 7 running at t = 98 (ticks in queue = 19) (in Q2).
PID 6 running at t = 99 (ticks in queue = 19) (in Q2).
PID 4 running at t = 99 (ticks in queue = 22) (in Q2).
PID 7 running at t = 99 (ticks in queue = 18) (in Q2).
PID 6 running at t = 100 (ticks in queue = 18) (in Q2).
[PROMOTE] PID 4 (RUNNING) promoted from level 2 to 0
[PROMOTE] PID 7 (RUNNING) promoted from level 2 to 0
PID 4 running at t = 100 (ticks in queue = 5) (in Q0).
[PROMOTE] PID 6 promoted from level 2 to 0
PID 7 running at t = 100 (ticks in queue = 5) (in Q0).
PID 7 running at t = 101 (ticks in queue = 4) (in Q0).
PID 4 running at t = 101 (ticks in queue = 4) (in Q0).
PID 6 running at t = 101 (ticks in queue = 5) (in Q0).
PID 7 running at t = 102 (ticks in queue = 3) (in Q0).
PID 4 running at t = 102 (ticks in queue = 3) (in Q0).
PID 6 running at t = 102 (ticks in queue = 4) (in Q0).
[PRIORITY] PID 7 (CPU Worker 2) at priority level 0, completed 600/1800 work units
PID 7 running at t = 103 (ticks in queue = 2) (in Q0).
```

The test results align with the design rules, confirming correct implementation of priority queue promotion, demotion, and starvation prevention.

3.4 Comparison with Original RR Scheduler

Metric	Original RR	3-level Queue
Response Time (PID 4)	0	1
Response Time (PID 5)	1	1
Response Time (PID 6)	1	1
Response Time (PID 7)	1	1
Response Time (PID 8)	1	1
Throughput	81	72.6

Table 1: Performance Comparison

3.5 Key Differences

- Response Time: Both have similar and small Response Time.

- Throughput: The 3-level queue scheduler provides smaller throughput.

3.6 Challenges and Solutions

1. Queue Management: Maintaining three separate queues while ensuring efficient process selection

   - Solution: Used simple link list approach with queue levels stored in process structures

2. Time Slice Tracking: Accurately tracking time slice usage across context switches

   - Solution: Used the existing timer interrupt mechanism with per-process tick counters

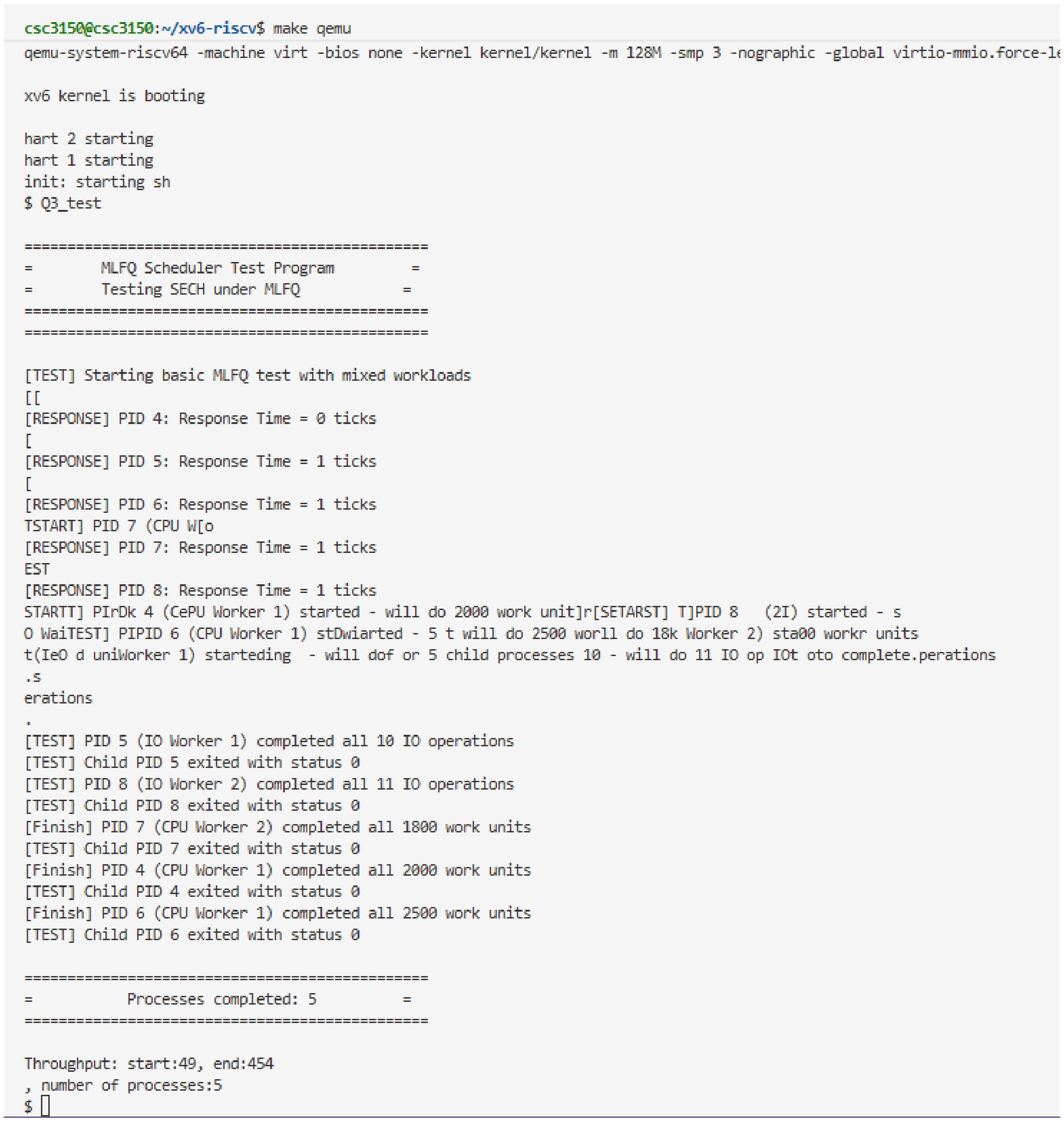
3. I/O Process Handling: Ensuring I/O-bound processes return to their original queues

   - Solution: Stored original queue information and restored it after I/O completion

4. Global Promotion Timing: Implementing the 100-tick promotion without interfering with normal scheduling

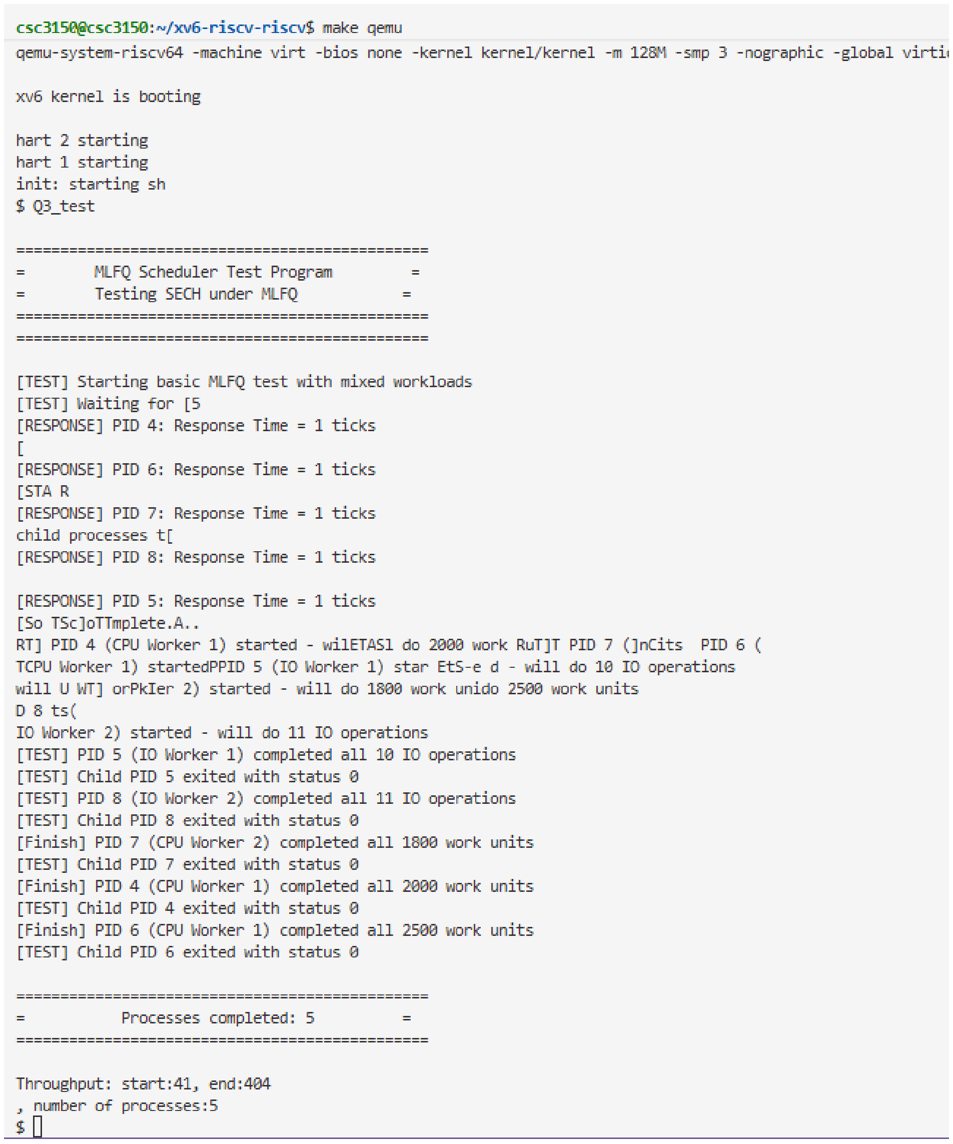
   - Solution: Used a global counter that increments on each timer interrupt

Note that I set 'MLFQ_DEBUG' as '0' and use same Q3_test.c(as submitted, but '#define MLFQ_DEBUG 0') in both part for fairness.



```
csc3150@csc3150:~/xv6-riscv$ make qemu
qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp 3 -nographic -global virtio-mmio.force-le

xv6 kernel is booting

hart 2 starting
hart 1 starting
init: starting sh
$ Q3_test

===============================================
=        MLFQ Scheduler Test Program         =
=        Testing SECH under MLFQ            =
===============================================
===============================================

[TEST] Starting basic MLFQ test with mixed workloads
[[
[RESPONSE] PID 4: Response Time = 0 ticks
[
[RESPONSE] PID 5: Response Time = 1 ticks
[
[RESPONSE] PID 6: Response Time = 1 ticks
TSTART] PID 7 (CPU W[o
[RESPONSE] PID 7: Response Time = 1 ticks
EST
[RESPONSE] PID 8: Response Time = 1 ticks
STARTT] PIrDk 4 (CePU Worker 1) started - will do 2000 work unit]r[SETARST] T]PID 8   (2I) started - s
O WaiTEST] PIPID 6 (CPU Worker 1) stDwiarted - 5 t will do 2500 worll do 18k Worker 2) sta00 workr units
t(IeO d uniWorker 1) starteding  - will dof or 5 child processes 10 - will do 11 IO op IOt oto complete.perations
.s
erations
.
[TEST] PID 5 (IO Worker 1) completed all 10 IO operations
[TEST] Child PID 5 exited with status 0
[TEST] PID 8 (IO Worker 2) completed all 11 IO operations
[TEST] Child PID 8 exited with status 0
[Finish] PID 7 (CPU Worker 2) completed all 1800 work units
[TEST] Child PID 7 exited with status 0
[Finish] PID 4 (CPU Worker 1) completed all 2000 work units
[TEST] Child PID 4 exited with status 0
[Finish] PID 6 (CPU Worker 1) completed all 2500 work units
[TEST] Child PID 6 exited with status 0

===============================================
=           Processes completed: 5          =
===============================================

Throughput: start:49, end:454
, number of processes:5
$ 
```

Original RR



```
csc3150@csc3150:~/xv6-riscv-riscv$ make qemu
qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp 3 -nographic -global virti

xv6 kernel is booting

hart 2 starting
hart 1 starting
init: starting sh
$ Q3_test

===============================================
=       MLFQ Scheduler Test Program          =
=       Testing SECH under MLFQ             =
===============================================
===============================================

[TEST] Starting basic MLFQ test with mixed workloads
[TEST] Waiting for [5
[RESPONSE] PID 4: Response Time = 1 ticks
[
[RESPONSE] PID 6: Response Time = 1 ticks
[STA R
[RESPONSE] PID 7: Response Time = 1 ticks
child processes t[
[RESPONSE] PID 8: Response Time = 1 ticks

[RESPONSE] PID 5: Response Time = 1 ticks
[So TSc]oTTmplete.A..
RT] PID 4 (CPU Worker 1) started - wilETAS1 do 2000 work RuT]T PID 7 (]nCits  PID 6 (
TCPU Worker 1) startedPPID 5 (IO Worker 1) star EtS-e d - will do 10 IO operations
will U WT] orPkIer 2) started - will do 1800 work unido 2500 work units
D 8 ts(
IO Worker 2) started - will do 11 IO operations
[TEST] PID 5 (IO Worker 1) completed all 10 IO operations
[TEST] Child PID 5 exited with status 0
[TEST] PID 8 (IO Worker 2) completed all 11 IO operations
[TEST] Child PID 8 exited with status 0
[Finish] PID 7 (CPU Worker 2) completed all 1800 work units
[TEST] Child PID 7 exited with status 0
[Finish] PID 4 (CPU Worker 1) completed all 2000 work units
[TEST] Child PID 4 exited with status 0
[Finish] PID 6 (CPU Worker 1) completed all 2500 work units
[TEST] Child PID 6 exited with status 0

===============================================
=          Processes completed: 5            =
===============================================

Throughput: start:41, end:404
, number of processes:5
$ 
```

3-level Queue.

4 Conclusion [2']

This assignment successfully implemented a 3-level priority queue scheduler for xv6-riscv that provides better responsiveness for interactive processes while maintaining fairness through anti-starvation mechanisms. The implementation demonstrates several important operating system concepts:

- Priority-based scheduling algorithms and their trade-offs
- Time slice management and process state tracking
- Anti-starvation mechanisms in priority schedulers
- Integration of new scheduling policies into existing OS kernels

The multi-level queue approach strikes a good balance between performance for interactive tasks and fairness for CPU-bound processes. The periodic promotion mechanism effectively addresses the starvation problem common in pure priority schedulers.